Rebatibilidad y Programación en Lógica Inductiva

Telma Delladio - Guillermo R. Simari

{td,grs}@cs.uns.edu.ar

Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA)

Departamento de Ciencias e Ingeniería de Computación (DCIC)

Universidad Nacional del Sur (UNS)

Avda. Alem 1253 - Bahía Blanca - Bs. As.

Tel (0291) 459-5135 - Fax (0291) 459-5136

Resumen

La Programación en Lógica Inductiva es una propuesta de aprendizaje de conceptos a partir de observaciones que utiliza Programación en Lógica como herramienta de representación de conocimiento y razonamiento. Una de las ventajas de las técnicas ILP es la expresividad del lenguaje utilizado (programación en lógica) y la capacidad de representar conocimiento previo, lo que es útil al momento de aprender nuevos conceptos o revisar conceptos ya existentes. Propuestas típicas en Programación en Lógica Inductiva utilizan paradigmas tradicionales de programación en lógica. Un problema que surge en estos casos es la falta de flexibilidad de la representación al momento de considerar nuevas observaciones, especialmente observaciones que representan casos particulares de los conceptos ya aprendidos. Mecanismos de razonamiento no monótono permitirían aprender definiciones más flexibles y adecuar las instancias particulares de estos conceptos en forma más natural. En este trabajo se propone la utilización de Programación en Lógica Rebatible como herramienta de representación de conocimiento y razonamiento no monótono a partir de la cual desarrollar aprendizaje inductivo de conceptos.

1 Introducción

La Programación en Lógica Rebatible (PLR) combina programación en lógica con argumentación rebatible. La PLR permite la representación de conocimiento a través de dos tipos de reglas; estrictas y rebatibles. Las *reglas rebatibles*, representan información débil o tentativa. Una regla rebatible de la forma $A \prec B$ es utilizada en la práctica para representar conocimiento rebatible. Esto es, conocimiento que puede considerarse válido solo si no hay razones fundadas para considerar lo contrario. Por ejemplo, $\sim$ *vuela* $\prec$ *mamífero*. Por otro lado, las *reglas estrictas* de la forma $X \leftarrow Y$ representan información no rebatible, como por ejemplo *cetáceo* $\leftarrow$ *delfín*.

Un programa lógico rebatible P es un par (K, R), donde K es el conjunto de información no rebatible representada por reglas estrictas y hechos lógicos, mientras que R representa el conocimiento rebatible a través de un conjunto de reglas rebatibles. La PLR ofrece un procedimiento de justificación que implementa un análisis dialéctico evaluando argumentos a favor y en contra de aquellas piezas de conocimiento bajo discusión. De esta forma un literal L estará bien justificado si existe un argumento para L que no puede ser derrotado en forma efectiva. En la PLR un argumento $\mathcal{A}$ para un literal p, $\langle \mathcal{A}, p \rangle$, es un conjunto minimal de reglas rebatibles consistente con el conocimiento estricto disponible (reglas estrictas y hechos) y que junto a este permite obtener una derivación para p. En este caso se dice que el argumento $\mathcal{A}$ soporta al literal p. Cualquier argumento formado únicamente por reglas rebatibles utilizadas en un argumento $\mathcal{A}$ se denomina subargumento de $\mathcal{A}$.

El análisis diálectico utilizado en la PLR considera todos los posibles argumentos en contra del argumento $\mathcal{A}$ que soporta p. Si existe algún argumento $\mathcal{B}$ que está en contra del argumento $\mathcal{A}$, y es lo suficientemente bueno, se dice que el argumento $\mathcal{B}$ derrota al argumento $\mathcal{A}$. Pero a su vez, si este argumento $\mathcal{B}$ es derrotado por un tercer argumento, $\mathcal{A}$ prevalecerá. El análisis dialéctico considerará entonces todos los derrotadores de $\mathcal{A}$ y luego los derrotadores de estos últimos y continuará con este procedimiento hasta determinar efectivamente si $\mathcal{A}$ prevalece o no. Para realizar este análisis se define la noción de *ataque* entre argumentos. Se dice que un argumento $\langle \mathcal{B}, q \rangle$ ataca a un argumento $\langle \mathcal{A}, p \rangle$, si existe un subargumento de $\mathcal{A}$, $\langle \mathcal{C}, r \rangle$, y se sabe que $\{q, r\}$ es inconsistente con el conocimiento estricto disponible. Dicho de otra forma, $\mathcal{B}$ soporta un literal (q) que va en contra de un literal (r) utilizado en la formación del argumento $\mathcal{A}$. La noción de ataque no es suficiente para determinar una relación de derrota entre argumentos. Es necesario tener un criterio de preferencia entre argumentos. Con estas dos nociones, ataque y preferencia entre argumentos se define la noción de *derrota*. Entonces, si existe un argumento $\mathcal{A}$ que es atacado en alguno de sus subargumentos $\mathcal{A}_{sub}$ por un argumento $\mathcal{B}$ y de acuerdo al criterio utilizado $\mathcal{B}$ es preferido a $\mathcal{A}_{sub}$, se dice que $\mathcal{B}$ es un *derrotador propio* de $\mathcal{A}$. En caso que ninguno de los argumentos considerados, $\mathcal{A}_{sub}$ y $\mathcal{B}$, sea preferido por sobre el otro, se dice que $\mathcal{B}$ *bloquea* a $\mathcal{A}$ y es un derrotador por bloqueo. En la PLR se utiliza *especificidad* [GS] como criterio de comparación entre argumentos. Intuitivamente, este criterio favorece a aquellos argumentos que utilizan mayor información del dominio y utilizan menos reglas. Dicho de otra forma, se prefieren aquellos argumentos que son más precisos y concisos.

2 Aprendizaje de conceptos

La propuesta de la Programación en Lógica Inductiva o ILP (por su denominación en inglés *Inductive Logic Programming*) se basa en el aprendizaje de conceptos a partir de ejemplos a través de la síntesis de programas lógicos que sirven como definición del concepto que se desea aprender.

Una definición general para el problema ILP considera como punto de partida un conjunto finito de reglas B, representando una base de conocimiento previa, y un conjunto de observa-

ciones. Estas observaciones pueden estar diferenciadas en observaciones positivas y negativas. El objetivo es encontrar un programa lógico P que junto con el conocimiento previo B sea *correcto* con respecto a las observaciones consideradas. Por lo general, esta noción de correctitud se basa en que $P \cup B$ cubra todas las observaciones positivas y no esté en contradicción con las observaciones negativas. Dependiendo las características en particular del problema considerado, se puede llevar adelante aprendizaje de uno o más conceptos a la vez. El procesamiento de las observaciones puede ser incremental o no. Las observaciones pueden ser positivas y negativas o solo positivas. El mecanismo de inducción utilizado puede ser top-down o bottom-up, *etc.* [NCd97]. Cualquiera sea el caso, la utilización de paradigmas de Programación en Lógica tradicionales hace que las definiciones obtenidas no se adapten fácilmente a nuevas observaciones. Para adaptar una definición a las nuevas observaciones puede ser necesario algún mecanismo de revisión de los conceptos ya aprendidos eliminando reglas aprendidas con anterioridad para conseguir una nueva definición *correcta* con respecto al nuevo conjunto de observaciones. Esta revisión puede ser costosa principalmente si se presentan como nuevas observaciones, casos particulares del concepto que se está aprendiendo.

2.1 PLR para el aprendizaje inductivo de conceptos

Una de las ventajas de la PLR en la representación de conocimiento es su capacidad para representar y razonar con información rebatible. Como toda propuesta de razonamiento no monótono, permite adaptar las conclusiones obtenidas al nuevo conocimiento disponible, lo que antes era un argumento puede dejar de serlo al considerar este nuevo conocimiento. Por eso la PLR se presenta como una herramienta interesante en la tarea del aprendizaje de conceptos a partir de observaciones. El aprendizaje de conceptos se lleva a cabo a través de la generación de definiciones para estos conceptos. Estas definiciones tendrán la forma de reglas rebatibles.

Esta propuesta, se basa entonces en la generación de reglas rebatibles que servirán en la formación de *argumentos* que garanticen las observaciones consideradas. En caso de existir observaciones que se estén modelando en forma incorrecta se deberá ajustar el conjunto de reglas rebatibles aprendidas para modelar las observaciones correctamente.

Ejemplo: Consideremos el siguiente programa lógico rebatible[1] $P = (K, \emptyset)$, donde el conjunto K está formado por los hechos

$$\begin{array}{ccc} \mathit{paloma(pio).} & \mathit{paloma(pepe).} & \mathit{gallina(tina).} \\ \mathit{gallina(turu).} & \mathit{bebé(pepe).} & \mathit{pingüino(tweety).} \\ & \mathit{asustada(tina).} & \end{array}$$

y las reglas

$$\begin{array}{lcllcl} \mathit{pájaro(X)} & \leftarrow & \mathit{paloma(X).} & \mathit{pájaro(X)} & \leftarrow & \mathit{gallina(X).} \\ \mathit{pájaro(X)} & \leftarrow & \mathit{pingüino(X).} & \sim\mathit{vuela(X)} & \leftarrow & \mathit{pingüino(X).} \end{array}$$

[1] Ejemplo adaptado de [GS]

Supongamos que se desea aprender el concepto *vuela* y para ello se cuenta con el siguiente conjunto de observaciones: $O = \{vuela(tina),\ vuela(pio),\ \sim vuela(turu),\ \sim vuela(tweety),\ \sim vuela(pepe)\ \}$

Si consideramos un mecanismo de inducción que comience con una definición general para luego ir especializándola a medida que se analizan las observaciones, se podrían considerar los siguientes pasos en la generación (aprendizaje) de reglas rebatibles, comenzando por la regla más general:

1. $vuela(X)$
2. $vuela(X) \prec pájaro(X)$.

   Esta regla es la especialización más simple a partir de la regla 1. Esta regla sirve como *argumento* de los literales $vuela(turu)$ y $vuela(pepe)$ lo cual va en contra de las observaciones. Se consideran entonces las reglas obtenidas a partir de esta última por aplicación de unfolding obteniendo las siguientes reglas:
3. $vuela(X) \prec paloma(X)$.
4. $vuela(X) \prec gallina(X)$.
5. $vuela(X) \prec pingüino(X)$.

   La regla 3 sirve como argumento para el literal $vuela(pepe)$ y no existen argumentos que lo derroten, pero se sabe que esto va en contra de la observación $\sim vuela(pepe)$. Es necesario entonces, modificar el conjunto de reglas rebatibles hasta aquí obtenidas de forma tal que no exista un argumento para $vuela(pepe)$ de tales características. Sin embargo, hay que notar que la regla 3 también sirve como argumento para la observación $vuela(pio)$, lo cual es correcto y dicho argumento no debería ser eliminado. Por lo tanto, se opta por refinar la regla 3 de forma tal que cubra correctamente la observación en cuestión, $\sim vuela(pepe)$. Para esto, se especializa el antecedente de la regla y se niega su consecuente obteniendo la regla:
6. $\sim vuela(X) \prec paloma(X), bebé(X)$.

   Esto permite que ahora, la regla 6 sirva como argumento para la observación $\sim vuela(pepe)$. Si bien es cierto que aún hay un argumento para el literal $vuela(pepe)$ (la regla 3), dicho argumento es derrotado por el argumento formado por la regla 6.

   Con respecto a la regla 4, se puede observar que dicha regla sirve como argumento para el literal $vuela(turu)$ que va en contra de la observación $\sim vuela(turu)$. Siguiendo la misma idea se podría refinar esta regla rebatible para adecuarla a esta observación. Como no hay conocimiento adicional disponible acerca del objeto *turu*, no se puede especializar el antecedente de la regla en forma natural. Por lo tanto, siguiendo la idea utilizada anteriormente, lo único que se hace es negar su consecuente. Con eso obtenemos la regla:
7. $\sim vuela(X) \prec gallina(X)$.

   Esta nueva regla sirve como argumento para la observación $\sim vuela(turu)$, pero este argumento no es mejor que el argumento dado por la regla 4 para el literal $vuela(turu)$. Ambos argumentos se *bloquean* y como sabemos que el literal $\sim vuela(turu)$ es cierto, (pues es una observación) optamos por eliminar del conjunto de reglas rebatibles la regla 4 para que prevalezca el argumento formado por la regla 7.

Ahora bien, la regla 7 sirve también como argumento para el literal $\sim$*vuela(tina)* lo cual va en contra de la observación *vuela(tina)*. Nuevamente especializamos el cuerpo de esta regla y negamos su consecuente, obteniendo una nueva regla rebatible:

8. $vuela(X) \prec gallina(X), asustada(X)$.
   Nuevamente, si bien es cierto que existe un argumento para el literal $\sim$*vuela(tina)* (la regla 7), este será derrotado por el argumento formado por la regla 8.

Así se concluye con la generación de reglas rebatibles para modelar el concepto *vuela*. A excepción de la regla 4 que se decidió eliminar, el resto de las reglas son consideradas como el resultado de la tarea de aprendizaje. Este conjunto de reglas rebatibles permite, junto con el conocimiento estricto dado originalmente garantizar cada una de las observaciones.

3 Conclusiones

La naturaleza no monótona de la PLR permite que el conjunto de literales sancionados pueda variar fácilmente con la modificación del conjunto de reglas rebatibles. Esto permite que la tarea de aprendizaje sea más flexible. Es necesario un estudio más acabado de esta propuesta, principalmente de la interacción entre las reglas rebatibles durante el proceso de aprendizaje. En el ejemplo se utilizaron básicamente las siguientes normas en la generación de reglas rebatibles:

- Si una de las reglas rebatibles aprendidas permite soportar únicamente observaciones en forma incorrecta se incluye la regla obtenida de negar su consecuente.
- Supongamos que existen dos argumentos $\mathcal{A}$ y $\mathcal{B}$ que se bloquean mutuamente y que soportan p y $\sim p$ respectivamente y se sabe además que p es una observación. En este caso, se elimina $\mathcal{B}$ (o parte de él) para permitir que el argumento $\mathcal{A}$ que soporta p prevalezca.
- Supongamos que existe una regla rebatible de la forma $A \prec B$ que justifica el literal p y es la única regla que permite la formación de un argumento que soporta q y se sabe además que $\sim p$ y q son observaciones. En este caso, se genera una nueva regla rebatible de la forma $\sim A \prec B, C$ que sirva en la formación de un argumento para $\sim p$. La especialización vía el literal C debe ser guiada por la observación $\sim p$ para no generar un argumento que derrote al argumento que utiliza $A \prec B$ para soportar q puesto que esta última modificación generará posiblemente un argumento más específico que el original.

Referencias

[GS] Alejandro Javier García and Guillermo Ricardo Simari. Defeasible logic programming: An argumentative approach. *To appear in Theory and Practice of Logic Programming.*

[NCd97] Shan-Hwei Nienhuys-Cheng and Ronald deWolf. *Foundations of Inductive Logic Programming.* Springer, 1997.